partial_report

Partial Report project AI Applications: AI4IM (simulated data Y2 labels)

Student: Alfredo Vargas

R-number: r0835034

Problem description

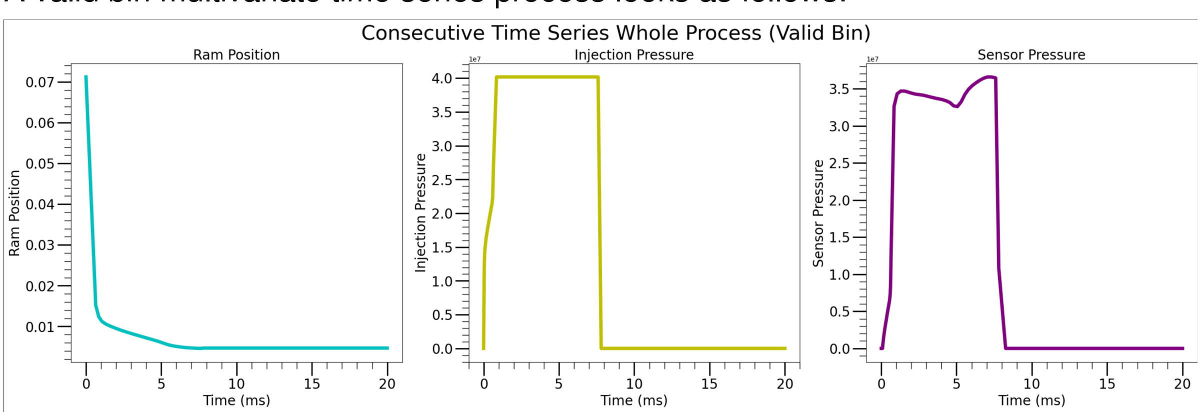
- Simulated Data of bin productions using Injection Moulding (Datapoints generated using Matlab)

Dataset

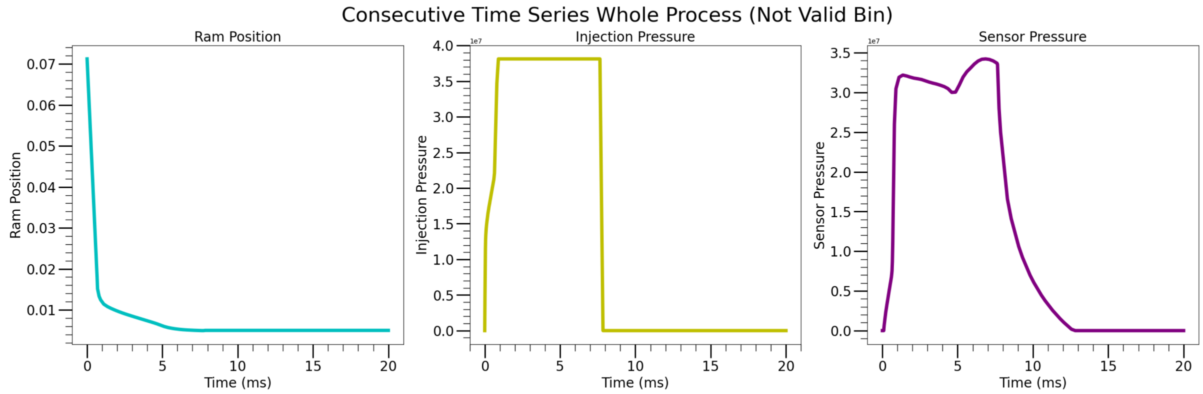
- 1542 datapoints
- The datapoints contains three time series:
  i. Ram position vs time
  ii. Injection Pressure vs time
  iii. Sensor Pressure vs time

Data Exploration

- A valid bin multivariate time series process looks as follows:



- A not valid bin multivariate time series process looks as follows:



- Although sometimes we see a differences between these two random selected points for valid and not valid. We cannot conclude that there is deterministic method to determine when a bin will have a defect or not. However, we see that there could be enough information to make statistical analysis, therefore ML methods can be suitable.

Labels:

- $Y_2 \in \mathbb{B}^{1542\times1}$, where $B = \{0, 1\}$ with 1 representing a valid bin and 0 a not valid bin.
- Description:

Feature	Dimension	Data Type
ramposition	1542	float
ramposition_time	1542	float
injection_pressure	1542	float
injection_pressure_time	1542	float
sensor_pressure	1542	float
sensor_pressure_time	1542	float

- Imbalanced dataset:
  - The number of valid bins is 1080 which correspond to the 70.04 %
  - The number of not valid bins is 462 which correspond to the 29.96 %

Goal:

- Get at least a performance of 80% for the f1-score for both the majority class (valid bins) and minority class (not valid bins).

Data preprocessing & Feature engineering

Feature engineering with `Helper.py` :

1. `series2features` function from `Helper.py` file, for each time-series generates 22 new engineered features.
2. After that concatenated features into one dataset matrix with dimension 66 features plus 1 label Y_2.

Feature engineering with `tsfresh` :

1. First trim the values **before** implementing feature engineering.
2. Concatenate the time series before implementing feature engineering with `tfresh` to incorporate the effects of a multivariate time series problem.
3. Select the most relevant features by specifying a `p-value` (This parameter was optimized!)

Data cleaning steps:

- Many of the engineered features have 0 constant values which after normalization become `NaN` values which are dropped.
- Some features have a constant value which can be dropped as they not contribute when one deals with some ML methods such a decision trees. However, we could keep it for its use when using other ML methods.
- To drop the features we use:

```
full_data_df.dropna(axis=1).describe()  # we drop along the columns axis=1
```

Pre-processing steps:

- Normalization done as follows: (standardization)

```
cols_to_norm = [i for i in range(0, 66)]  # we exclude the labels
```

```
full_data_df[cols_to_norm] = full_data_df[cols_to_norm].apply(lambda x: (x
full_data_df.head()
```

Data splits, tried the following:

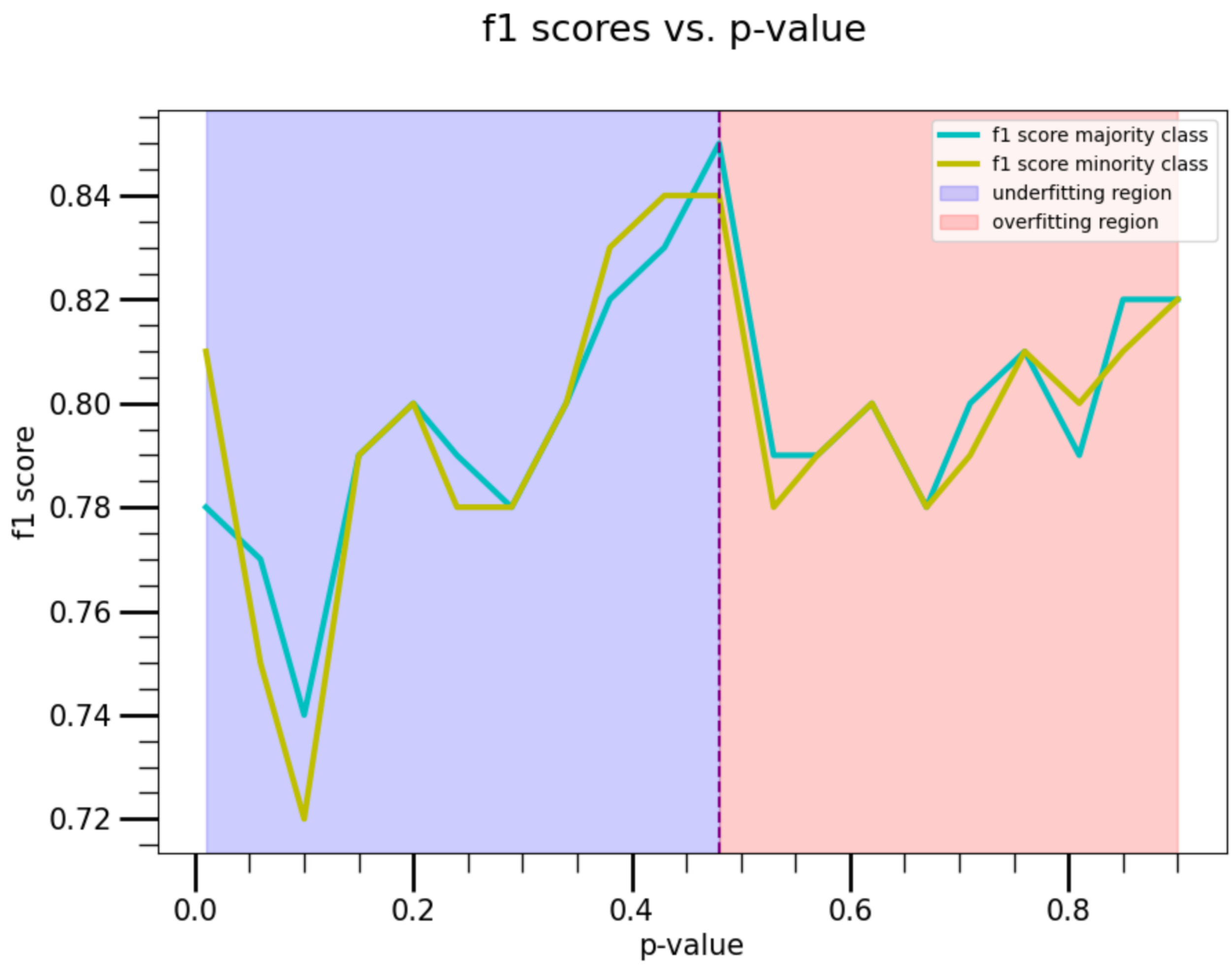
1. Split 80% for train and 20% for test, validation not included as Random Forest incorporates validation given by the number of estimators (number of trees).
2. $p - value$ tunned to determine how many features must be kept in order to obtain the best results (f1 scores).
3. TODO: explore validation splits when using other ML methods.

Methods

1. Random Forest Classifier
2. Neural Networks
   - TODO
3. SVM
   - TODO

Results

1. Random Forest:
   - **p-value threshold** for feature selection when using `tsfresh` was chosen by optimizing the `f1score` of both minority and majority classes when using the defaults parameters of Random Forest classifier provided by `scikit-learn`. The optimal `p-value` found has a value of 0.48. See figure below:



Conclusions

- So far we have a good `f1-score` for both the minority and majority classes with values above 80%.
- We can clearly identified two regions when feature engineering one related to the under fitting region when `p-values` when using the `tfresh`. One region corresponds to the under-fitting region ($p - value < 0.48$), meaning we have less features (52% or more are considered rare features and therefore ignored). The other region corresponds to the over-fitting region with $p - value > 0.48$ (52% or less are considered rare features and therefore ignored.)
- TODO:
  - Explore cross validation with other ML methods.
  - Obtain optimal `p-values` when using other ML methods.
  - Explore **both** strategy for data augmentation when using the `tsfresh` library. So far we have used only the `minority` strategy for data augmentation.